# Refine Search

## Search Results -

| Term | Documents |
|---|---|
| "7284009" | 2 |
| 7284009S | 0 |
| "7284009".PN..PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD. | 1 |
| (7284009.PN.).PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD. | 1 |

**Database:**

```
US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins
```

**Search:**

L28

Refine Search

Recall Text       Clear       Interrupt

## Search History

**DATE: Thursday, October 25, 2007**      **Purge Queries**      Printable Copy      Create Case

| Set Name side by side | Query | Hit Count | Set Name result set |
|---|---|---|---|
| | *DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR* | | |
| L28 | 7284009.pn. | 1 | L28 |
| | *DB=USPT; PLUR=YES; OP=OR* | | |
| L27 | 7284009.pn. | 1 | L27 |
| L26 | L25 and (unique near n-gram$) | 0 | L26 |
| L25 | I22 and (frequency) | 2 | L25 |
| L24 | I22 and (low near frequency) | 0 | L24 |
| L23 | I22 and (low near frquency) | 0 | L23 |
| L22 | L21 and n-grams | 3 | L22 |
| L21 | L20 and characters | 31 | L21 |
| L20 | cluster$ near strings | 73 | L20 |
| L19 | cluster$ near (plurality near strings) | 1 | L19 |

| L18 | cluster$ near (plurtality near strings) | 0 | L18 |
|------|------|------|------|
| L17 | I14 and (classif$ or cluster$ or group$) | 5 | L17 |
| L16 | I14 and (high ) | 4 | L16 |
| L15 | I14 and (high near frequency) | 0 | L15 |
| L14 | L13 and pair | 5 | L14 |
| L13 | L12 and (unique near n-gram) | 5 | L13 |
| L12 | (low near frequency) and (n-gram) | 55 | L12 |
| L11 | (low near frequency) and (pair near n-gram) | 0 | L11 |
| L10 | L9 and high | 1 | L10 |
| L9 | L8 and pair | 1 | L9 |
| L8 | L7 and string | 1 | L8 |
| L7 | I6 and n-gram$ | 1 | L7 |
| L6 | low near frequency near n-gram$ | 1 | L6 |

*DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR*

| L5 | 20050022111.pn. | 2 | L5 |
|------|------|------|------|

*DB=USPT; PLUR=YES; OP=OR*

| L4 | (6006221.pn.) and (n-gram$) | 1 | L4 |
|------|------|------|------|
| L3 | (6006221.pn.) and (unique near n-gram$) | 0 | L3 |
| L2 | L1 and (unique near n-gram$) | 7 | L2 |
| L1 | pair same (frequency near n-gram$) | 11 | L1 |

END OF SEARCH HISTORY

┌   Generate Collection     Print

L6: Entry 1 of 1                          File: USPT                Aug 7, 2007


DOCUMENT-IDENTIFIER: US 7254774 B2
TITLE: Systems and methods for improved spell checking

PRIOR-PUBLICATION:
DOC-ID                          DATE
US 20050210383 A1               September 22, 2005


Description Paragraph (73):
On the other hand, query logs can be quite unreliable for low-count unigrams and
bigrams. For example, 20 queries in a sample query log contained the bigram catfish
soap while only 3 contained the bigram catfish soup. Based on these statistics, a
query such as catfish sop would be associated with the incorrect alternative
containing the word soap (for simplicity, this example assumes that dist(sop, soap)
=dist(sop, soup)). In one instance of the present invention, word unigrams and
bigrams that appear in the query logs are employed, but have their query-log
frequencies adjusted according to their web frequencies. In this way, very low
frequency n-grams can be filtered out from the query log that do not occur on the
web (this can be done to limit the size of the data utilized at runtime), but do
not lose higher frequency query misspellings that are useful for iterative
correction, while obtaining more reliable word n-gram statistics. In FIG. 5, an
illustration of an information flow structure 500 in accordance with an aspect of
the present invention is shown. The information flow structure 500 depicts a web
index 502 being utilized to re-estimate unigram and bigram statistics for a query
log trie 504.

Description Paragraph (79):
Looking at FIG. 8, yet another flow diagram of a method 800 of facilitating search
queries in accordance with an aspect of the present invention is illustrated. The
method 800 starts 802 by obtaining web statistics for low-count query log n-grams
from an inverted web index 804. This enhances the statistical information for the
n-gram by incorporating information from a much larger data base. The web
statistics are then utilized, at least in part, as the statistics for the n-gram
for iterative processing of low-count query log n-grams 806, ending the flow 808.
In one instance of the present invention, word unigrams and bigrams that appear in
the query logs are employed, but have their query-log frequencies adjusted
according to their web frequencies. In this way, very low frequency n-grams can be
filtered out from the query log that do not occur on the web (this can be done to
limit the size of the data utilized at runtime), but do not lose higher frequency
query misspellings that are useful for iterative correction, while obtaining more
reliable word n-gram statistics.

L14: Entry 1 of 5                    File: USPT              Oct 16, 2007


DOCUMENT-IDENTIFIER: US 7284009 B2
TITLE: System and method for command line prediction

PRIOR-PUBLICATION:
DOC-ID                          DATE
US 20040117380 A1               June 17, 2004


Abstract Text (1):
Systems, methods, and computer program products for command line prediction are
disclosed. Commands entered on a command line are saved to a command log. The
command log is analyzed to generate a plurality of n-gram tables that reflect the
entries in the command log. The n-gram tables may be stored in a suitable memory.
Subsequent command sequences entered on the command line may be compared to the n-
gram tables to assess the likelihood of a command.

Brief Summary Text (7):
In one aspect, a method for command line prediction in a computer system is
provided. The computer system includes a database of command sequences organized
into tables of n-grams. The method comprises receiving an entry in a command line
sequence; searching the database for an n-gram entry matching the received entry in
the command line sequence; and displaying a predicted command if an n-gram entry
that satisfies a certainty threshold is found in the database.

Brief Summary Text (8):
In another aspect, a method for building a database for use in command line
prediction is provided. The method comprises storing commands entered by a user in
a command log; generating a plurality of n-gram tables from the commands stored in
the command log, wherein the n-gram tables include an indicator of the frequency
with which an n-gram appears in the command log; and storing the n-gram tables in
memory.

Brief Summary Text (9):
In another aspect, a computer program product for command line prediction in a
computer system is provided. The computer system includes a database of command
sequences organized into tables of n-grams. The computer program product comprises
logic instructions, executable on a processor, for receiving an entry in a command
line sequence; logic instructions, executable on a processor, for searching the
database for an n-gram entry matching the received entry in the command line
sequence; and logic instructions, executable on a processor, for displaying a
predicted command if an n-gram entry that satisfies a certainty threshold is found
in the database.

Brief Summary Text (10):
In another aspect, a system for command line prediction is provided. The system
comprises a processor configured to collect commands input by a user and store the
commands in a command log; a processor configured to retrieve entries from the
command log and to generate a plurality of n-gram tables from the commands stored
in the command log, wherein the n-gram tables include an indicator of the frequency
with which an n-gram appears in the command log; a database for storing the n-gram

tables; and a processor configured to receive an entry in a command line sequence
input by a user, to search the database for an n-gram entry matching the received
entry in the command line sequence; and to display a predicted command if an n-gram
entry that satisfies a certainty threshold is found in the database.

Description Paragraph (15):
In addition, a sequence of commands can be viewed as an n-gram. For example, the
command cd mydir may be viewed as an n-gram of length 1, also referred to as a 1-
gram. The command sequence cd mydir, ls-alt, may be viewed as an n-gram of length
2, i.e., a 2-gram. Similarly, the command sequence cd mydir, ls-alt, emacs myfile
can be viewed as three tokens in an n-gram of length 3, i.e., a 3-gram, and so on
and so forth.

Description Paragraph (22):
The Background process 120 periodically analyzes the command log(s) and the session
log to populate a series of data tables of n-grams that may be stored in the
database 140. The first table simply stores the total number of command sequences
(i.e., the sum of each n-gram*the frequency of the n-gram). For the example entries
considered above, and assuming the system is configured to collect n-grams up to
length 5, the table may look as follows:

Description Paragraph (24):
Background process 120 also populates data tables in database 140 that store the
various n-grams recorded in the command log and the corresponding frequency with
which the n-grams are used. In an exemplary embodiment, these data tables may be
designed using a relational model that implements separate tables for each length
of n-gram. Thus each table may be for an n-gram of length X, and may contain a
foreign key to the table for n-grams of length X-1. For example, the table for 5-
grams may include a single entry for each n-gram and a foreign key pointing to the
table for the 4-gram that is a subset of the 5-gram. Similarly, the table for 4-
grams may include a single entry and a foreign key pointing to the table for the 3-
gram that is a subset of the 4-gram. In like fashion the table for 3-grams may
refer to the table for 2-grams, which may in turn refer to the table for 1-grams.
One of skill in the art will appreciate that the table for 1-grams does not need a
foreign key.

Description Paragraph (33):
In addition, the system implements an in-memory data structure containing up to the
last MAX_LENGTH commands processed by the system. In an exemplary embodiment, the
in-memory data structure may be configured as a FIFO data buffer 210, as depicted
in FIG. 2. For example, if the system is configured to process 5-grams, then the
in-memory data structure 210 is configured to hold 5 entries 212a-212e from the
command log. Background process 120 may read entries from the command log into the
FIFO data buffer 210 in a reverse chronological order, such that the most recent
entry in the command log is read first. Each time a new entry is read from the
command log into the FIFO data buffer 210, the entries in command log 210 are
shifted to the left, such that space 212a always holds the most recent entry read
from the command log, space 212b always holds the second most recent entry from the
command log, space 212c always holds the third most recent entry read from the
command log, space 212d always holds the fourth most recent entry read from the
command log, and space 212e always holds the fifth most recent entry read from the
command log. In addition, each time a new entry is read from the command log into
the FIFO buffer 210, background process 120 updates the various n-gram tables in
database 140, a process described in detail below.

Description Paragraph (34):
In addition, foreground process 110 uses an in-memory data structure that is a
collection of matches from the n-gram DB. This consists of a sequence of commands
(the n-gram) and the frequency of the n-grams from the DB. This data structure may
be a list of n-gram, frequency pairs. An exemplary entry in the data structure may

be as follows:

Description Paragraph (35):
TABLE-US-00010 N-Gram Frequency (cd mydir,ls-alt,emacs myfile) 1

Description Paragraph (40):
FIG. 3 is a flowchart illustrating an exemplary background process 120. This
process may execute periodically or may wait until a session has been recorded in
the command log and the session log before execute. Among other things, background
process 120 analyzes the command log file described above to populate the various
n-gram tables. In an exemplary embodiment, background process may retrieve the
contents of a single session. This may be accomplished by first retrieving the
login time and logout time of a session, which may then be deleted from the
database. All of the entries entered in the command log during the session may be
analyzed together.

Description Paragraph (41):
FIG. 3 is a schematic illustration of a background process 120 in which the system
is configured to match n-grams up to length 5. Referring to FIG. 3, at step 300 an
entry is read from the command log. At step 312 the entry is entered into the first
space 212a of FIFO data buffer 210 depicted in FIG. 2. At step 316 background
process 120 searches the 1-gram data table in the database 140 for an entry that
matches the most recent entry 212a in the FIFO data buffer 210. If no match is
found at step 320, then background process 120 adds the most recent entry 212a in
the FIFO data buffer 210 as a new entry in the 1-gram table at step 324. By
contrast, if a match is found at step 320 then background process 120 updates the
frequency counter(s) for the matching entry in the 1-gram table at step 328.

Description Paragraph (46):
For example, the first entry is written to the 1-gram table, the second entry is
written to the 2-gram table and a foreign key is set to the entry in 1-gram table,
the third entry is written to the 3-gram table and a foreign key is set to the
entry in the 2-gram table, the fourth entry is written to the 4-gram table and a
foreign key is set to the entry in the 3-gram table, and the fifth entry is written
to the 5-gram table and a foreign key is set to the entry in the 4-gram table. One
of skill in the art will appreciate that a longer FIFO buffer could be used to
record N-grams having a length greater than 5. In general, the FIFO buffer may be
dimensioned to record up to MAX_LENGTH n-grams from length 1 to MAX_LENGTH i.e. 1
n-gram of each length.

Description Paragraph (47):
One of skill in the art will recognize that the system need not be limited to n-
grams of any particular length. In practice, the system may be implemented to match
n-grams up to any arbitrary length, MAX_LENGTH. Database 140 may be updated to
include an n-gram table for n-grams of length i, where i is incremented from 1 to
MAX-LENGTH. FIFO data buffer 210 may be implemented as a buffer of length
MAX_LENGTH. One of skill in the art will also recognize that background process 120
may be implemented as an iterative process of searching the n-gram tables for
matching n-grams of length i, where i is incremented from 1 to MAX_LENGTH.

Description Paragraph (49):
The database comprising the n-grams may be flushed periodically. Flushing, in the
data structure given above, simply means deleting records from each n-gram table
that have a very low frequency count (below a certain threshold). Over long periods
of time, a user will enter a lot of unique n-grams that may never contribute to a
recommendation. These do not need to be stored in the database. In addition, data
that is older than a given threshold may be flushed from the system.

Description Paragraph (51):
In an exemplary embodiment, foreground process 110 logs the commands entered by the

user to the command log, searches the n-gram tables generated by the background
process for potential matches, implements a likelihood algorithm to predict likely
next commands, and recommend commands to the users. In addition, foreground process
110 maintains an in-memory data structure that is a listing of n-grams matching
those entered by the user and the frequency with which the n-grams occur in the
data tables. Foreground process 110 may be incorporated into the command shell. It
will be appreciated that if a user opens multiple shells, then each shell may
execute an independent foreground process, and that each foreground process may
generate and maintain independent command logs.

Description Paragraph (54):
At step 525 the various n-gram tables in the database may be scanned for matching
n-grams, and at step 530 foreground process stores the highest matching n-gram(s)
and their associated frequency in memory. At step 535 foreground process executes a
likelihood algorithm that assesses the likelihood that the n-gram(s) are predictive
of the next command to be entered by the user. At step 540 foreground process 110
displays the command(s) that satisfy a likelihood threshold.

Description Paragraph (55):
In an exemplary embodiment, steps 525 and 530 may be implemented as an iterative
search process that traverses the n-gram tables stored in the database searching
for the highest-order n-gram that matches the commands stored in the FIFO data
buffer 210 used by foreground process 110 and saves the matching n-gram (or
information sufficient to identify it) and a frequency indicator that identifies
the number of times the matching n-gram was entered in the n-gram tables.

Description Paragraph (57):
At step 610 a command is received, e.g., as when a user enters a command on the
command line. At step 614 the FIFO data buffer is updated, e.g., by writing the
received command into element 1 of the FIFO data buffer and shifting the existing
contents one element in the array. At step 618, counters i and j are set to 1. At
step 622 the 1-gram table is searched for 1-grams matching element j of the FIFO
data buffer. If there is no match in the 1-gram table (step 626), then control
passes back to step 610 and foreground process 110 waits for the next command to be
entered. By contrast, if there is a match in the 1-gram table (step 626), then i is
incremented at step 630 and control passes to step 634, wherein the i-gram table is
searched for i-grams starting with element j of the FIFO data buffer. If there is
no match in the i-gram table (step 638), then control passes back to step 610 and
foreground process 110 waits for the next command to be entered. By contrast, if
there is a match at step 638, then information identifying the i-gram and its
frequency indicator are stored in a memory at step 642. At step 646 i is compared
to MAX_LENGTH, and if i is greater than or equal to MAX_LENGTH, then control passes
back to step 610 and foreground process 110 waits for the next command to be
entered. By contrast, if i is less than MAX_LENGTH, then i and j are incremented
(step 650) and control passes to step 654. Where the FIFO date buffer is examined
to determine whether there is another command in the queue to process. If there is
another command in the queue, then control passes to Step 634, and the process
checks the next-highest i-gram table. This iterative process traverses the n-gram
tables populated by background process 120 to find the highest-order n-grams
matching the commands stored in the FIFO data buffer.

Description Paragraph (58):
FIG. 6 will now be discussed in greater detail for purposes of clarity. Assuming
that the system has been configured to analyze n-grams of length 5, the background
process will populate tables for 1-grams, 2-grams, 3-grams, 4-grams, and 5-grams in
the database 140. Foreground process 110 may be configured to traverse some or all
of the n-gram tables to generate predictions for likely next commands.

Description Paragraph (60):
When the user enters the second command, the command is received at step 610, and

at step 614 the FIFO data buffer is updated, e.g., by shifting the first command
one space and writing the second command into the FIFO data buffer. At step 618 i
and j are initialized to 1. At step 622 the 1-gram table is searched for 1-grams
corresponding to the first entry in the FIFO data buffer, i.e., Command.sub.1. If
no match is found (step 622) then control passes back to step 610 and the
foreground process 110 waits for the next command to be entered. By contrast, if a
match is found in the 1-gram table then i is incremented (step 630) to 2 and at
step 634 the 2-gram table is searched for entries that correspond to the first
entry in the FIFO data buffer. These entries represent a 2-gram having the form of
{Command.sub.1, X}, where X is a wildcard. If there are no matches in the 2-gram
table, then control passes back to step 610. By contrast, if there are one or more
matches, then at step 642 the matching 2-grams and their corresponding frequency
values are written to an in-memory data structure maintained by the foreground
process. At step 646 the counter i, the value of which is 2, is less than
MAX_LENGTH so control passes to step 650, which increments i to 3 and j to 2. At
step 654 there is a Command.sub.2 in the FIFO data buffer, so control passes to
step 634, where the 3-gram table is searched for 3-grams having the form
{Command.sub.2, Command.sub.1, X}, where X is a wildcard. In an exemplary
embodiment, this search may be performed by first searching the 3-gram table for
all entries beginning with Command.sub.2, then selecting from this set the entries
having a foreign key to an entry in the 2-gram table having the format
{Command.sub.1, X}, resulting in one or more entries from the 3-gram table having
the form {Command.sub.2, Command.sub.1, X}. If there were no matches in the 3-gram
table (step 638), then control passes back to step 610. By contrast, if there was a
match, then the in-memory data table is updated at step 642. In an exemplary
embodiment, foreground process 110 logs the highest-order n-gram match. Therefore,
the matching entries from the 3-gram table having the form {Command.sub.2,
Command.sub.1, X} may be written over the matches from the 2-gram table having the
form {Command.sub.1, X}. At step 646 i, which hold the value of 3, is less than
MAX_LENGTH so control passes to step 650 and i is incremented to a value of 4 and j
is incremented to a value of 3. At step 654 there is no command in the third
element of the FIFO data buffer so control passes to step 610.

Description Paragraph (65):
An exemplary command line prediction system employs an algorithm to assess the
likelihood the next command based on the most recent command(s) input from the
user. A command likelihood algorithm may take a list of n-gram matches with their
frequencies, and calculate the likelihood of each match being correct for the next
command. An exemplary algorithm may calculate the likelihood of an n-gram as
proportional to it's frequency divided by the total number of n-grams of that
length. Expressed as a formula, the algorithm may determine: L=a*F/T

Description Paragraph (67):
L=Likelihood of the n-gram.

Description Paragraph (68):
F=Frequency of the n-gram.

Description Paragraph (69):
T=Total number of n-grams of this length.

Description Paragraph (72):
Alternate embodiments of the likelihood algorithm may implement additional
predictive features. For example, the likelihood threshold may be adjusted based on
the length of the n-gram. In an exemplary embodiment, the likelihood threshold may
be inversely proportional to the length of the n-gram match. This accommodates for
the fact that the likelihood of finding a matching n-gram in the data tables is
also inversely proportional to the length of the n-gram. For example, in any search
there should be many more 2-grams matches than 5-gram matches.

Description Paragraph (73):
In another embodiment, the system may ignore commands that are separated by an
amount of time that exceeds a threshold. This recognizes that the predictive value
of a command being entered is affected by the time between the command being
entered and the previous command being entered. For example, if a user enters two
commands `quickfire`, then they are likely related. By contrast, if the user pauses
for 30 minutes between commands, then perhaps the system should not consider this
as an n-gram, and instead start from scratch with a 1-gram on the second command
being entered. The time threshold may be set by the system designer or by the user
of the system as a parameter.

Description Paragraph (74):
In addition, if the user ends his or her session, then the system may also end its
session. When a user's session ends, the background process may treat this as the
end of any n-grams taken from the command log.

CLAIMS:

1. A method for command line prediction in a computer system including a database
of command sequences organized into tables of n-grams, comprising: receiving an
entry in a command line sequence; searching the database for an n-gram entry
matching the received entry in the command line sequence; and displaying a
predicted command if an n-gram entry that satisfies a certainty threshold is found
in the database.

4. The method of claim 1, wherein searching the database for an n-gram entry
matching the received entry in the command line sequence comprises searching for a
1-gram entry that matches the received entry.

9. The method of claim 1, further comprising determining a likelihood that the
matching n-grams are predictive of the next command wherein the likelihood is
inversely proportional to the length of the n-gram match.

10. The method of claim 1, further comprising determining a likelihood that the
matching n-grams are predictive of the next command wherein the likelihood is a
function of the frequency of the matching n-grams and the total number of matching
n-grams of that corresponding length.

11. A computer program product for command line prediction in a computer system
including a database of command sequences organized into tables of n-grams,
comprising: logic instructions, executable on a processor, for receiving an entry
in a command line sequence; logic instructions, executable on a processor, for
searching the database for an n-gram entry matching the received entry in the
command line sequence; and logic instructions, executable on a processor, for
displaying a predicted command if an n-gram entry that satisfies a certainty
threshold is found in the database.

14. The computer program product of claim 11, wherein logic instructions for
searching the database for an n-gram entry matching the received entry in the
command line sequence comprise logic instructions for searching for a 1-gram entry
that matches the received entry.

19. The computer program product of claim 11, further comprising determining logic
instructions for a likelihood that the matching n-grams are predictive of the next
command wherein the likelihood is inversely proportional to the length of the n-
gram match.

20. The computer program product of claim 11, further comprising determining logic
instruction for a likelihood that the matching n-grams are predictive of the next
command wherein the likelihood is a function of the frequency of the matching n-

grams and the total number of matching n-grams of that corresponding length.

21. A system for command line prediction, comprising: a processor configured to collect commands input by a user and store the commands in a command log; a processor configured to retrieve entries from the command log and to generate a plurality of n-gram tables from the commands stored in the command log, wherein the n-gram tables include an indicator of the frequency with which an n-gram appears in the command log; a database for storing the n-gram tables; and a processor configured to receive an entry in a command line sequence input by a user, to search the database for an n-gram entry matching the received entry in the command line sequence; and to display a predicted command if an n-gram entry that satisfies a certainty threshold is found in the database.

22. The system of claim 21, wherein the processor is configured to search the database for an n-gram entry matching the received entry in the command line sequence comprises searching for a 1-gram entry that matches the received entry.

27. The system of claim 21, further comprising a processor configured to determine a likelihood that the matching n-grams are predictive of the next command wherein the assigned likelihood is inversely proportional to the length of the n-gram match.

28. The system of claim 21, further comprising determining logic instruction for a likelihood that the matching n-grams are predictive of the next command wherein the likelihood is a function of the frequency of the matching n-grams and the total number of matching n-grams of that corresponding length.

L25: Entry 1 of 2                    File: USPT                Jun 10, 2003


DOCUMENT-IDENTIFIER: US 6578032 B1
TITLE: Method and system for performing phrase/word clustering and cluster merging

<u>Abstract Text</u> (1):
Text classification has become an important aspect of information technology.
Present text classification techniques range from simple text matching to more
complex clustering methods. Clustering describes a process of discovering structure
in a collection of <u>characters</u>. The invention automatically analyzes a text string
and either updates an existing cluster or creates a new cluster. To that end, the
invention may use a <u>character n-gram</u> matching process in addition to other
heuristic-based clustering techniques. In the <u>character n-gram</u> matching process,
each text string is first normalized using several heuristics. It is then divided
into a set of overlapping <u>character n-grams,</u> where n is the number of adjacent
<u>characters</u>. If the commonality between the text string and the existing cluster
members satisfies a pre-defined threshold, the text string is added to the cluster.
If, on the other hand, the commonality does not satisfy the pre-defined threshold,
a new cluster may be created. Each cluster may have a selected topic name. The
topic name allows whole clusters to be compared in a similar way to the individual
clusters, and merged when a predetermined level of commonality exists between the
subject clusters. The topic name also may be used as a suggested alternative to the
text string. In this instance, the topic name of the cluster to which the text
string was added may be outputted as an alternative to the text string.

<u>Brief Summary Text</u> (9):
Text classification has become an important aspect of information technology.
Present text classification techniques range from simple text matching to more
complex clustering methods. Clustering describes a process of discovering structure
in a collection of <u>characters</u>. The invention automatically analyzes a text string
and either updates an existing cluster or creates a new cluster. To that end, the
invention may use a <u>character n-gram</u> matching process in addition to other
heuristic-based clustering techniques. In the <u>character n-gram</u> matching process,
each text string is first normalized using several heuristics. It is then divided
into a set of overlapping <u>character n-grams,</u> where n is the number of adjacent
<u>characters</u>. If the commonality between the text string and the existing cluster
members satisfies a pre-defined threshold, the text string is added to the cluster.
If, on the other hand, the commonality does not satisfy the pre-defined threshold,
a new cluster may be created. Each cluster may have a selected topic name. The
topic name allows whole clusters to be compared in a similar way to the individual
<u>clusters or strings,</u> and merged when a predetermined level of commonality exists
between the subject clusters. The topic name also may be used as a suggested
alternative to the text string. In this instance, the topic name of the cluster to
which the text string was added may be outputted as an alternative to the text
string.

<u>Brief Summary Text</u> (10):
More specifically, the invention provides a method, system and computer-readable
medium having computer-executable instructions for clustering <u>character</u> strings.
Each <u>character</u> string comprises a word or a phrase. The method comprises the steps
of receiving at least one <u>character string, and clustering</u> a first <u>character</u> string
with another <u>character</u> string into one or more groups, when the first <u>character</u>

string satisfies a predetermined degree of commonality with one or more character strings in each of these groups. When the first character string does not satisfy the predetermined level of commonality with another character string, another group is created. The method also selects at least one of the character strings in each of the groups to be the group's topic name. Selection of the topic may be based on a pre-designation or a frequency of the received character strings with the groups. The selected topic may then be outputted.

Detailed Description Text (16):
FIG. 2 is a block diagram of a client-server system in which the present invention may be implemented. Client-server system 200 includes a client computer 201 coupled to a communication network 205. Client computer 201 may comprise a personal computer (as shown in FIG. 1) that has an extensible markup language (xml) and/or hypertext markup language (html)-based browser software installed thereon, for example, INTERNET EXPLORER available from MICROSOFT Corporation. Communication network 205 may be a LAN or WAN, for example, the Internet. It should be understood that while one client computer 201 is shown in FIG. 2, in practice, there may be many client computers simultaneously accessing communication network 205. Communication network 205 is further coupled to a search engine server 204. Search engine server 204 is coupled to a search engine database 211. Client computer 201 sends a query 202 to search engine server 204 via communication network 205. In return from search engine server 204, client computer 201 receives a search result 203, corresponding to data located in search engine database 211. Query 202 may contain various combinations of characters, for example, alphanumeric or ASCII entities. It should be appreciated that query 202 also may include non-alphanumeric, graphic-based entities including, but not limited to, bit-mapped graphic images. It should also be appreciated that although the following description uses examples with lowercase alphanumeric characters, the present invention may be capable of discerning lowercase with uppercase alphanumeric characters.

Detailed Description Text (22):
Each cluster 306-309 may also designate at least one of its members to be a topic name. A topic name is one or more words or phrases that describe all members! of the cluster. Selection of a particular topic may be based on any number of factors including, but not limited to, the frequency with which a particular member is entered as a query and a predetermined user designation. In the example shown in FIG. 3, "pokemon" 300 is the topic for cluster A 306 because it is the only member of cluster A 306. However, if another of cluster A's 306 members, for example "pokeman" 301, was queried by users more often than "pokemon" 300, "pokeman" 301 may become the topic for cluster A 306. Alternatively, a database manager may predetermine that "pokemon" 300 will remain the topic for cluster A 306, regardless of the frequency of other queries. Selection of the topic will be discussed further with reference to FIG. 10.

Detailed Description Text (23):
As received queries 301-304 enter QCluster computer 207, QClustering program 305 compares the members of clusters 306-309 with received queries 301-304 to determine which cluster will house each query 301-304. QClustering program 305 may compare received query 301-304 with the members of clusters 306-309 using any number of techniques. FIGS. 4-7 show one such method for comparing received queries 301-304 to the contents of editorial database 210, called "bigram matching." Bigram matching is a technique that segments each word or phrase to be compared into a plurality of character sets. Each character set includes two adjacent characters of the subject word. For example, as shown in FIG. 4, received query "pokeman" 301 has the following character sets: "_p"; "po"; "ok"; "ke"; "em"; "ma"; "an"; "n_";

Detailed Description Text (24):
Notably, the bigram character sets include spaces (i.e., "_") at the beginning and end of each word. This bigram segmenting is accomplished for received queries 301-

304, as well as members of clusters 306-309. Although FIGS. 4-7 illustrate the
comparison of received queries 301-304 with the members of clusters 306-309 using
bigram matching, it should be appreciated that any n-gram matching may be
conducted, for example, trigram or quadgram. It should also be appreciated that the
invention may conduct the comparison of received queries 301-304 with the members
of clusters 306-309 using other matching techniques.

Detailed Description Text (26):
FIG. 4 is an example comparison of a received query with a member of a cluster
using bigram matching, according to the invention. Specifically, FIG. 4 shows the
comparison of received query "pokeman" 301 with topic "pokemon" 300, where other
received queries 302-304 have not yet been considered by QClustering program 305.
As shown in FIG. 4, received query "pokeman" 301 is divided into bigram 401 and
compared with bigram 402 of topic "pokemon" 300, located in cluster A 306. Matching
character sets of bigram 402 are shown highlighted. The highlighted matching
character sets of bigram 402 include "_p", "po", "ok", "ke", "em", "n_". Once
bigram 401 for received query "pokeman" 301 is compared with bigram 402 of topic
"pokemon" 300, a bigram match score 403 maybe determined. Here, bigram match score
403 for bigram 401 and bigram 402 has a value of 6/8. This means that six of a
possible eight character sets of bigram 402 matched bigram 401 for received query
"pokeman" 301. Although the bigram match score has been described as a fraction of
matching bigrams within the entire domain, it should be appreciated that other
scoring techniques may be used to determine the bigram match score.

Detailed Description Text (28):
In addition to weighting the bigram match score based on the length of the query or
cluster member, the bigram match score may be weighted based on characteristics of
the individual character sets. This weighting recognizes that certain bigram
character sets appear less than others, and thus likely are more significant and
should be given greater weight. For example, in FIG. 4, matching character set "ke"
may be so rare that its very presence signals that the query should be clustered
with the matching member. This weighting may be accomplished in QCluster computer
207 by QClustering program 305. Although two examples of weighting were discussed,
it should be appreciated that there may be many other weighting techniques, based
on the characteristics of the query or cluster members.

Detailed Description Text (42):
In step 1004, QCluster Program 305 may calculate the frequency of the occurrence of
the individual words and whole query. In step 1005, the highest frequency words and
queries are determined, based on step 1004. The precise number of selected highest
frequency "items" (i.e., words and/or queries) may vary, depending on the relative
scores. For example, the two highest frequency items may be selected when their
frequency scores are relatively close. On the other hand, only one highest
frequency item may be selected, where the subject item has a frequency score that
is significantly higher than the second highest frequency item. If two or more
highest frequency items are selected, it is determined whether the items have the
same frequency score, in step 1006. If the scores are not the same, the highest
frequency item may be selected as the topic. Alternatively, a predetermined number
of highest frequency items may be selected to be the topics. If the highest
frequency items have the same frequency score, a predetermined criterion may be
used to break the tie, in step 1008. For example, it may be that the longest item
(i.e., the item with the most characters) is selected as the topic. Notably, if
none of the items satisfy a predetermined minimum threshold to become a topic, it
may be that the longest item is selected to be the topic of the cluster.

Detailed Description Text (43):
The invention is directed to a system and method for classifying a character string
from database entries, but is not limited to database information, regardless of
any specific description in the drawing or examples set forth herein. Moreover, it
should be appreciated that the invention is not limited to clustering information

anew, but also may be adapted to merging existing clusters of information. It will be understood that the present invention is not limited to use of any of the particular components or devices herein. Indeed, this invention can be used in any application that requires the categorization of words or phrases, including spell-checking software, for example. Further, the system disclosed in the present invention can be used with the method of the present invention or a variety of other applications.

CLAIMS:

2. The method of claim 1, wherein said selection of said first and said another topic name further comprise determining a frequency of said words or phrases in said clusters, wherein said first and said another topic names satisfy a predetermined level of frequency in said clusters.

3. The method of claim 1, wherein said comparing further comprises segmenting said first topic name into a first plurality of character sets and said another topic name into another plurality of character sets, and comparing said first plurality of character sets with said another plurality of character sets.

4. The method of claim 3, wherein each of said character sets comprise more than one adjacent characters of said character string.

5. A method for classifying information, comprising: receiving at least one character string, wherein each character string comprises a word or a phrase; segmenting a first character string into a first plurality of character sets and a another character string into another plurality of character sets, wherein each of said character sets comprise more than one adjacent characters of said character string; comparing said first plurality of character sets with said another plurality of character sets; clustering said first character string with said another character string into a group, when said first character set satisfies a predetermined degree of commonality with said another character set; creating another group when said first character set does not satisfy said predetermined degree of commonality with said another character set; selecting at least one of said character strings in each of said groups to be a topic, based on a frequency of said character strings with said groups; and outputting said topic.

6. The method of claim 5, wherein said character string is received via a communication network.

8. The method of claim 5, wherein said character sets comprise combinations of two adjacent characters.

9. The method of claim 5, wherein said character sets include a blank character at a beginning of said word or phrase.

10. The method of claim 5, wherein said character sets include a blank character at an end of said word or phrase.

11. The method of claim 5, wherein said comparing further comprises comparing each character set in said first plurality of character sets to each character set in said second plurality of character sets.

12. The method of claim 5, wherein said predetermined degree of commonality is a function of the number of matching character sets between said first character set and said second character set.

16. The method of claim 5, wherein said topic is a predetermined character string.

17. The method of claim 5, wherein said topic is a character string that is

received more than other character strings.